EXPRESS MAIL NO. [EL696998573US]

DETECTING COMPROMISED BALLOTS

RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/270,182 filed February 20, 2001, claims the benefit of U.S. Provisional Application No. ___________ (patent counsel's docket number 32462-8006US02) filed February 11, 2002, and is a continuation-in-part of each of U.S. Patent Application No. 09/534,836, filed March 24, 2000; U.S. Patent Application No. 09/535,927, filed March 24, 2000; and U.S. Patent Application No. 09/816,869 filed March 24, 2001. Each of these five applications is incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention is directed to the fields of election automation and cryptographic techniques therefor.

BACKGROUND

[0003] The problems of inaccuracy and inefficiency have long attended conventional, manually-conducted elections. While it has been widely suggested that computers could be used to make elections more accurate and efficient, computers bring with them their own pitfalls. Since electronic data is so easily altered, many electronic voting systems are prone to several types of failures that are far less likely to occur with conventional voting systems.

[0004] One class of such failures relates to the uncertain integrity of the voter's computer, or other computing device. In today's networked computing environment, it is extremely difficult to keep any machine safe from malicious software. Such software is often able to remain hidden on a computer for long periods of time before actually performing a malicious action. In the meantime, it may replicate itself to other computers on the network, or computers that have some minimal interaction with the network. It may even be transferred to computers that are not networked by way of permanent media carried by users.

[0005] In the context of electronic secret ballot elections, this kind of malicious software is especially dangerous, since even when its malicious action is triggered, it may go undetected, and hence left to disrupt more elections in the future. Controlled logic and accuracy tests ("L&A tests") monitor the processing of test ballots to determine whether a voting system is operating properly, and may be used in an attempt to detect malicious software present in a voter's computer. L&A tests are extremely difficult to conduct effectively, however, since it is possible that the malicious software may be able to differentiate between "real" and "test" ballots, and leave all "test" ballots unaffected. Since the requirement for ballot secrecy makes it impossible to inspect "real" ballots for compromise, even exhaustive L&A testing may prove futile. The problem of combating this threat is known as the "Client Trust Problem."

[0006] Most existing methods for solving the Client Trust Problem have focused on methods to secure the voting platform, and thus provide certainty that the voter's computer is “clean,” or “uninfected.” Unfortunately, the expertise and ongoing diligent labor that is required to achieve an acceptable level of such certainty typically forces electronic voting systems into the controlled environment of the poll site, where the client computer systems can be maintained and monitored by computer and network experts. These poll site systems can still offer some advantages by way of ease of configuration, ease of use, efficiency of tabulation, and cost. However, this approach fails to deliver on the great potential for distributed communication that has been exploited in the world of e-commerce.

[0007] Accordingly, a solution to the Client Trust Problem that does not require the voting platform to be secured against malicious software, which enables practically any computer system anywhere to be used as the voting platform, would have significant utility.

BRIEF DESCRIPTION OF DRAWINGS

[0008] Figure 1 is a high-level block diagram showing a typical environment in which the facility operates.

[0009] Figure 2 is a block diagram showing some of the components typically incorporated in at least some of the computer systems and other devices on which the facility executes.

[0010] Figure 3 is a flow diagram showing steps typically performed by the facility in order to detect a compromised ballot.

DETAILED DESCRIPTION

[0011] A software facility for detecting ballots compromised by malicious programs ("the facility") is provided. The approach employed by the facility typically makes no attempt to eliminate, or prevent the existence of malicious software on the voting computer. Instead, it offers a cryptographically secure method for the voter to verify the contents of the voter's ballot *as it is received at the vote collection center*, without revealing information about the contents (ballot choices) to the collection center itself. That is, the vote collection center can confirm to the voter exactly what choices were received, without knowing what those choices are. Thus, the voter can detect any differences between the voter's *intended* choices, and the *actual* choices received at the vote collection center (as represented in the transmitted voted ballot digital data). Further, each election can choose from a flexible set of policy decisions allowing a voter to re-cast the voter's ballot in the case that the received choices differ from the intended choices.

[0012] The facility is described in the context of a fairly standard election setting. For ease of presentation, initial discussion of the facility assumes that there is only one question on the ballot, and that there are a set of K allowable answers, $a_1,\ldots,a_K$ (one of which may be "abstain"). It will be appreciated by those of ordinary skill in the art that it is a straightforward matter to generalize the solution given in this situation to handle the vast majority of real world ballot configurations.

[0013] Several typical cryptographic features of the election setting are:

1. ***Ballot Construction:*** A set of cryptographic *election parameters* are agreed upon by election officials in advance, and made publicly known by wide publication or other such means. Significant parameters are the *encryption*

group, *generator*, *election public key* and *decision encoding scheme*. More specifically, these are:

(a) The **encryption group**, G may be Z_p, with p a large prime, or an elliptic curve group.

(b) The **generator**, $g \in G$. In the case $G=Z_p$, g should generate a (multiplicative) subgroup, $\langle g \rangle$, of G^* which has large prime order q. In the elliptic curve case we assume $\langle g \rangle = G$ and $q=p$.

(c) The **election public key**, $h \in \langle g \rangle$.

(d) The **decision encoding scheme**: A partition of $\langle g \rangle$ into "answer representatives." That is, $\langle g \rangle = S_0 \cup S_1 \cup \cdots S_K$, where the S_k are pair wise disjoint subsets of $\langle g \rangle$. For each $1 \leq k \leq K$, any message $m \in S_k$ represents a vote for a_k. The remaining messages, $m \in S_0$ are considered *invalid*. Typically, each S_k, $1 \leq k \leq K$, consists of a single element, μ_k, though this is not, fundamentally, a requirement. For the security of the scheme, however, it is generally required that the μ_k are generated *independently at random* either using some public random source, or by an acceptable sharing scheme.

[0014] While the following discussion uses multiplicative group notation for the sake of consistency, it should be clear that all constructions can be implemented equally well using elliptic curves.

2. ***Vote Submission***: Each voter, v_i, *encrypts* her vote, or decision, as an ElGamal pair, $(X_i, Y_i) = (g^{\alpha_i}, h^{\alpha_i}, m_i)$, where $\alpha_i \in Z_q$ is chosen randomly by the voter, and $m_i \in S_k$ if v_i wishes to choose answer a_k. This encrypted value is what is transmitted to the vote collection center (cast), usually with an attached digital signature created by v_i.

[0015] If the voter, v_i, were computing these values herself - say with pencil and paper - this protocol would essentially suffice to implement a secret ballot, universally verifiable election system. (Depending on the tabulation method to be used, some additional information, such as a voter proof of validity would be necessary.) However, since in practice, v_i only makes choices through some user interface, it is not realistic to expect her to observe the actual value of the bits sent and check them for consistency with her intended choice. In short, the vote client can ignore voter intent and submit a "μ_j vote" when the voter actually wished to submit a "μ_k vote."

[0016] The voter typically needs some way to verify that the encrypted vote which was received at the vote collection center is consistent with her choice. Simply making the ballot box data public does not a reasonable solution, since the vote client, not the voter, chooses α_i. For reasons of vote secrecy, and coercion, this value should be "lost." So v_i's encrypted vote is as opaque to her as it is to anyone else. A generic confirmation from the vote collection center is obviously not sufficient either. The general properties of what is needed are properties:

1. The confirmation string, C, returned by the vote collection center, needs to be a function of the data (encrypted vote) received.

2. The voter and vote client should be able to execute a specific set of steps that allow the voter to tie C exclusively to the choice (or vote), μ_k, *that was received.*

3. It should be impossible for the vote client to behave in such a way that the voter "is fooled." That is, the client can not convince the voter that μ_k was received, when actually, $\mu \neq \mu_k$ was received.

[0017] In this section, we present such a scheme, which we shall refer to as *SVC*, in its basic form. In following sections, we offer some improvements and enhancements.

[0018] The following steps are typically performed as part of the voting process.

CC-1. The vote client, M_i, "operated by" v_i, creates an encrypted ballot on behalf of v_i as before. Let us denote this by $(X_i, Y_i) = (g^{\alpha_i}, h^{\alpha_i} m_i)$, for some value $m_i \in \langle g \rangle$ and $\alpha_i \in Z_q$.

CC-2. M_i is also required to construct a validity proof, P_i, which is a zeroknowledge proof that $m_i \in \{\mu_1, \ldots, \mu_K\}$. (Such a proof is easily constructed from the basic Chaum-Pederson proof for equality of discrete logarithms using the techniques of [CDS94]. See [CGS97] for a specific example.)

CC-3. M_i then submits both P_i and the (signed) encrypted vote, (X_i, Y_i) to the vote collection center.

CC-4. Before accepting the encrypted ballot, the vote collection center first checks the proof, P_i. If verification of P_i fails, corruption has already been detected, and the vote collection center can either issue no confirmation string, or some default random one.

CC-5. Assuming then that verification of P_i succeeds, the vote collection center computes the values, W_i and U_i as,

$$W_i = K_i Y_i^{\beta_i} = K_i h^{\alpha_i \beta_i} m_i^{\beta_i} \quad (1)$$

$$U_i = h^{\beta_i} \quad (2)$$

where $K_i \in G$ and $\beta_i \in Z_q$ are generated randomly and independently (on a voter-by-voter basis).

CC-6. The vote collection center then returns (U_i, W_i) to M_i.

CC-7. The client, M_i, computes

$$C_i = V_i / U_i^{\alpha_i} = K_i m_i^{\beta_i} \quad (3)$$

and display this string (or, more likely, a hash of it, $H(C_i)$) to the voter, v_i.

[0019] The voter needs to know which confirmation string to look for. This can be accomplished in two different ways. The most straightforward is to have the voter, v_i, obtain K_i and β_i from the vote collection center. This is workable, requires very little data to be transferred, and may be well suited to some implementations. However, in other situations, it may be an unattractive approach because C_i (or $H(C_i)$) must then be computed. Since asking M_i to perform this computation would destroy the security of the scheme, v_i must have access to an additional computing device, as well as access to the independent communication channel.

[0020] An alternative is to have the vote collection center compute all possible confirmation strings for v_i, and send what amounts to a *confirmation dictionary* to v_i via the independent channel. In general, the confirmation dictionary for voter v_i would consist of the following table laid out in any reasonable format:

Answer	Confirmation String
a_1	$H(C_{i1})$
a_2	$H(C_{i2})$
$\vdots$	$\vdots$
a_K	$H(C_{iK})$

where H is the election's public (published) hash function (possibly the identity function), and $C_{ij}=K_i\mu_j^{\beta_i}$.

[0021] Of course care must be used in engineering the independent channel to be sure that it really is independent. Ideally, it should be inaccessible to devices connected to the voting network. Solutions are available, however. Since the K_i and β_i can be generated in advance of the election, even slow methods of delivery, such as surface mail, can be employed to transmit the dictionary.

[0022] In order to more completely describe the facility, an example illustrating the operation of some of its embodiments is described. The following is a detailed example of a Secret Value Confirmation exchange.

[0023] In order to maximize the clarity of the example, several of the basic parameters used – for example, the number of questions on the ballot, and the size of the cryptographic parameters – are much smaller than those that would be typically used in practice. Also, while aspects of the example exchange are discussed below in a particular order, those skilled in the art will recognize that they may be performed in a variety of other orders.

[0024] Some electronic election protocols include additional features, such as:

- voter and authority certificate (public key) information for authentication and audit
- ballot page style parameters
- data encoding standards

- tabulation protocol and parameters

[0025] As these features are independent of the Secret Value Confirmation implementation, a detailed description of them is not included in this example.

[0026] This example assumes an election protocol that encodes voter responses (answers) as a single ElGamal pair. However, from the description found here, it is a trivial matter to also construct a Secret Value Confirmation exchange for other election protocols using ElGamal encryption for the voted ballot. For example, some embodiments of the facility incorporate the homomorphic election protocol described in U.S. Patent Application No. 09/535,927. In that protocol, a voter response is represented by multiple ElGamal pairs. The confirmation dictionary used in this example is easily modified to either display a concatenation of the respective confirmation strings, or to display a hash of the sequence of them.

[0027] The jurisdiction must first agree on the election initialization data. This at least includes: the basic cryptographic numerical parameters, a ballot (i.e., a set of questions and allowable answers, etc.) and a decision encoding scheme. (It may also include additional data relevant to the particular election protocol being used.)

Cryptographic Parameters

- Group Arithmetic: Integer multiplicative modular arithmetic
- Prime Modulus: $p = 47$
- Subgroup Modulus: $q = 23$
- Generator: $g = 2$
- Public Key: $h = g^s$ where s is secret. For the sake of this example, let us say that $h = g^{12} = 7$.

Ballot

- One Question

– Question 1 Text: *Which colors should we make our flag? (Select at most 1.)*

– Number of answers/choices: 4

* Answer 1 Text: *Blue*

* Answer 2 Text: *Green*

* Answer 3 Text: *Red*

* Answer 4 Text: *I abstain*

Decision Encoding Scheme

Choice	Response Value
Blue	$9(\mu_1)$
Green	$21(\mu_2)$
Red	$36(\mu_3)$
I abstain	$17(\mu_4)$

[0028] At some point, before issuing a confirmation *and* before distributing the voter confirmation dictionaries, the ballot collection center (or agency) generates random, independent β_i and K_i for each voter, V_i. If the confirmation dictionary is to be sent after vote reception, these parameters can be generated, on a voter by voter basis, immediately after each voted ballot is accepted. Alternatively, they can be generated in advance of the election. In this example, the ballot collection agency has access to these parameters both immediately after accepting the voted ballot, and immediately before sending the respective voter's confirmation dictionary.

[0029] Sometime during the official polling time, each voter, V, obtains and authenticates the election initialization data described above. It can be obtained by submitting a "ballot request" to some ballot server. Alternatively, the jurisdiction may have some convenient means to "publish" the election initialization data – that is, make it conveniently available to all voters.

[0030] From the election initialization data, V is able to determine that the expected response is the standard encoding of a particular sequence of *two* distinct data elements. These are (in their precise order):

Choice Encryption

[0031] A pair of integers (X, Y) with $0 \leq X, Y < 47$ indicating (in encrypted form) the voter's choice, or answer. For the answer to be valid, it must be of the form, $(X, Y) = (2^{\alpha}, 7^{\alpha}\mu)$, where $0 \leq \alpha < 23$ and $\mu \in \{9, 21, 36, 17\}$.

Proof of Validity

[0032] A *proof of validity* showing that (X, Y) is of the form described in the choice encryption step above. (In this example, we shall see that this proof consists of 15 modular integers arranged in specific sequence.)

[0033] For the sake of this example, let us assume that V wishes to cast a vote for "Green."

1. V generates $\alpha \in Z_{23}$ randomly. In this example, $\alpha = 5$. Since the encoding of "Green" is 21, V's choice encryption is computed as

$$(X, Y) = (2^5, 7^5 \times 21) = (32, 24) \qquad (4)$$

   This pair is what *should* be sent to the vote collection center. The potential threat is that V's computer may try to alter these values.

[0034] Voter V (or more precisely, V's computer) must prove that *one* of the following conditions hold

1. $(X, Y) = (2^{\alpha}, 7^{\alpha} \times 9)$ i.e. choice (vote cast) is "Blue"

2. $(X, Y) = (2^{\alpha}, 7^{\alpha} \times 21)$ i.e. choice (vote cast) is "Green"

3. $(X,Y)=(2^{\alpha}, 7^{\alpha}\times 36)$ i.e. choice (vote cast) is “Red”

4. $(X,Y)=(2^{\alpha}, 7^{\alpha}\times 17)$ i.e. choice (vote cast) is “I abstain”

for some unspecified value of α without revealing *which* of them actually does hold.

[0035] There are a variety of standard methods that can be used to accomplish this. See, for example, R. Cramer, I. Damgård, B. Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Advances in Cryptology - CRYPTO ’94, Lecture Notes in Computer Science, pp. 174-187, Springer-Verlag, Berlin, 1994. The Secret Value Confirmation technique used by the facility works equally well with any method that satisfies the abstract criteria of the previous paragraph. While details of one such validity proof method are provided below, embodiments of the facility may use validity proofs of types other than this one.

Validity Proof Construction:

[0036] (In what follows, each action or computation which V is required to perform is actually carried out by V's computer.)

1. V sets $\alpha_2 = \alpha = 5$.

2. V generates $\omega_2 \in_R Z_{23}$, $r_1, r_3, r_4 \in_R Z_{23}$, $s_1, s_3, s_4 \in_R Z_{23}$ all randomly and independently. For this example we take

$$\omega_2 = 4 \qquad (5)$$

$$r_1 = 16, \quad r_3 = 17, \quad r_4 = 21$$

$$s_1 = 12, \quad s_3 = 4, \quad s_4 = 15$$

3. V computes corresponding values

$$a_1 = g^{r_1}X^{-s_1} \quad = \quad 2^{16}\times 32^{11} = 4 \qquad (6)$$

$$a_2 = g^{\omega_2} \quad = \quad 2^4 = 16$$

$$a_3 = g^{r_3} X^{-s_3} \quad = \quad 2^{17} \times 32^{19} = 6$$

$$a_4 = g^{r_4} X^{-s_4} \quad = \quad 2^{21} \times 32^{8} = 9$$

$$b_1 = h^{r_1} (Y/9)^{-s_1} \quad = \quad 7^{16} \times (24/9)^{11} = 18$$

$$b_2 = h^{\omega_2} \quad = \quad 7^{4} = 4$$

$$b_3 = h^{r_3} (Y/36)^{-s_3} \quad = \quad 7^{17} \times (24/36)^{19} = 1$$

$$b_4 = h^{r_4} (Y/17)^{-s_5} \quad = \quad 7^{21} \times (24/17)^{8} = 7$$

(7)

4. V uses a publicly specified hash function H to compute $c \in Z_{23}$ as

$$c = H\left(\{X, Y, a_i, b_i\}\right) \quad 1 \le i \le 4 \qquad (8)$$

Since many choices of the hash function are possible, for this example we can just pick a random value, say

$$c = 19. \qquad (9)$$

(In practice, SHA1, or MD5, or other such standard secure hash function may be used to compute H.)

5. V computes the interpolating polynomial $P(x)$ of degree $4-1=3$. The defining properties of P are

$$P(0) = c = 19 \qquad (10)$$

$$P(1) = s_1 = 12$$

$$P(3) = s_3 = 4$$

$$P(4) = s_4 = 15$$

$P(x)=\sum_{j=0}^{3} z_j x^j$ is computed using standard polynomial interpolation theory, to yield:

$$P(x)=x^3+20x^2+18x+19 \tag{11}$$

or

$$\begin{matrix} z_0=19 & z_1=18 \\ z_2=20 & z_3=1 \end{matrix} \tag{12}$$

6. V computes the values

$$s_2=P(2)=5 \tag{13}$$

$$r_2=\omega_2+\alpha_2 s_2=4+5\times 5=6$$

7. V's validity proof consists of the 12 numbers

$$\{a_k, b_k, r_k\}_{k=1}^{6} \tag{14}$$

and the three numbers

$$\{z_k\}_{k=1}^{3} \tag{15}$$

in precise sequence. (z_0 need not be submitted since it is computable from the other data elements submitted using the public hash function H.)

[0037] Having computed the required choice encryption, (X, Y), and the corresponding proof of validity, V encodes these elements, in sequence, as defined by the standard encoding format. The resulting sequences form V's voted ballot. (In order to make the ballot unalterable, and indisputable, V may also digitally sign this voted ballot with his private signing key. The resulting combination of V's voted ballot, and his digital signature (more precisely, the standard encoding of these two elements) forms his signed voted ballot.) Finally, each voter transmits his (optionally signed) voted ballot back to the data center collecting the votes.

[0038] As described above, the voter specific random parameters for V $(\beta$ and $K)$ are available at the vote collection center. In this example, these are

$$\beta = 18 \qquad K = 37 \tag{16}$$

[0039] When the voter's (optionally signed) voted ballot is received at the vote collection center, the following steps are executed

1. The digital signature is checked to determine the authenticity of the ballot, as well as the eligibility of the voter.

2. If the signature in step 1 verifies correctly, the vote collection center then verifies the proof of validity. For the particular type of validity proof we have chosen to use in this example, this consists of

   (a) The public hash function H is used to compute the value of $P(0) = z_0$

   $$z_0 = P(0) = H\left(\{X, Y, a_i, b_i\}_{i=1}^{4}\right) = 19 \tag{17}$$

   (Recall that the remaining coefficients of P, z_1, z_2, z_3, are part of V's (optionally signed) voted ballot submission.)

   (b) For each $1 \le j \le 4$ both sides of the equations

   $$a_j = g^{r_j} x_j^{-P(j)} \tag{18}$$

   $$b_j = h^{r_j} \left(y_j / \mu_j\right)^{-P(j)}$$

   are evaluated. (Here, as described above, the μ_j are taken from the Decision Encoding Scheme.) If equality fails in *any* of these, verification fails. This ballot is not accepted, and some arbitrary rejection string (indication) is sent back to V.

3. Assuming that the previous steps have passed successfully, the reply string (W, U) is computed as

$$W = KY^{\beta} = 37 \times 24^{18} = 9 \tag{19}$$

$$U = h^{\beta} = 7^{18} = 42$$

This sequenced pair is encoded as specified by the public encoding format, and returned to V.

4. V's computer calculates

$$C = W/U^{\alpha} = 9/(42)^5 = 18 \tag{20}$$

and displays this string to V. (Alternatively, the protocol may specify that a public hash function is computed on C and the resulting hash value displayed. In this example, C itself is displayed.) If V's computer attempted to submit a choice other than "Green," the value of C computed above would be different. Moreover, the correct value of C cannot be computed from an incorrect one without solving the Diffie-Hellman problem. (For the small values of p and q we have used here, this is possible. However, for "real" cryptographic parameters, V's computer would be unable to do this.) Thus, if V's computer has submitted an encrypted ballot which does not correspond to V's choice, there are only two things it can do at the point it is expected to display a confirmation. It can display something, or it can display nothing. In the case that nothing is displayed, V may take this as an indication that the ballot was corrupted. In the case that something is displayed, what is displayed will almost certainly be wrong, and again, V may take this as an indication that the ballot was corrupted.

5. V now compares the value of C displayed to the value found in V's confirmation dictionary corresponding to the choice, "Green" (V's *intended* choice). At this point, V may have already received his confirmation dictionary in advance, or may obtain a copy through any independent channel. An example of such a channel would be to use a fax machine. If the displayed value does not match the corresponding confirmation string in

the confirmation dictionary, corruption is detected, and the ballot can be "recast" in accordance with election-specific policy.

[0040] Each voter confirmation dictionary is computed by the vote collection center, since, as described above, it is the entity which has knowledge of the voter specific values of α and K. For the case of the voter, V, we have been considering, the dictionary is computed as

Choice	Confirmation String
"Blue"	$C_1 = K\mu_1^{\beta} = 37 \times 9^{18} = 16$
"Green"	$C_2 = K\mu_2^{\beta} = 37 \times 21^{18} = 18$
"Red"	$C_3 = K\mu_3^{\beta} = 37 \times 36^{18} = 36$
"I abstain"	$C_4 = K\mu_4^{\beta} = 37 \times 17^{18} = 8$

[0041] The level of security provided by the facility when using the SVC scheme is described hereafter: Let A be the vote client adversary, and let $\in_0$ be an upper bound on the probability that A is able to forge a validity proof for any given $\mu_1,\ldots,\mu_K$. (We know that $\in_0$ is negligible.)

[0042] **Theorem 1** *Suppose the SVC scheme is executed with $H = Id$. Fix $1 \leq k_1 \neq k_2 \leq K$. Suppose that for some $\in > 0$, A can, with probability $\in$, submit $b_i = \left(g^{\alpha_i}, h^{\alpha_i}\mu_{k_1}\right)$, and display $C_{ik_2} = K_i\mu_{k_2}^{\beta_i}$, where the probability is taken uniformly over all combinations of values for $\mu_1,\ldots,\mu_K$, g, h, β_i* and *K_i. Then A can solve a random instance of the Diffie-Hellman problem with probability $\in$, and with $O(K)$ additional work.*

[0043] **Proof:** Suppose A is given $X,Y,Z \in_R \langle g\rangle$. A can simulate an election and SVC exchange by picking $C_{ik1} \in \langle g\rangle$ and $\mu_k \in \langle g\rangle$ independently at random *for* all $k \neq k_2$, setting $h = X, h^{\beta_i} = Y$ and $\mu_{k2} = \mu_{k1}Z$. The resulting distribution on the election parameters and C_{ik_1} is obviously identical to the distribution that arises from real elections. With probability $\in$, A can display C_{ik_2}, so can compute

$$C = C_{ik_2}/C_{ik_1} = \left(\mu_{k_2}/\mu_{k_1}\right)^{\beta_i} = Z^{\beta_i} \qquad (20)$$

So $\log_X C = \beta_i \log_h Z = \log_X Y \log_X Z$, and C is the solution to the Diffie-Hellman problem instance posed by the triple (X,Y,Z).

[0044] **Corollary 1** *Suppose again that the SVC scheme is executed with $H = Id$. Fix $1 \geq k_2 \geq K$. Suppose that for some $\in_1 > 0$, A can, with probability $\in_1$, choose $k_1 \neq k_2$, submit b_i = $\left(g^{\alpha_i}, h^{\alpha_i} \mu_{k_1}\right)$, and displays $C_{ik_2} = K_i \mu_{k_2}^{\beta_i}$, where the probability is taken uniformly over all combinations of values for $\mu_1,\ldots,\mu_K$, g, h, β_i and K_i. Then A can solve a random instance of the Diffie-Hellman problem with probability $\in_1/(K\text{-}1)$, and with O(K) additional work.*

[0045] **Proof:** Follow the arguments of theorem 1, but compare to the problem of finding the solution to at least one of K-1 independent Diffie-Hellman problems.

[0046] **Corollary 2** *Let $\in_{DH}$ be an upper bound on the probability that A can solve a random Diffie-Hellman instance. Then, in the case that H = Id, an upper bound on the probability that A can submit a vote that differs from the voter's choice, and yet display the correct confirmation string is $\in_0 + (K-1)\in_{DH}$.*

[0047] If the hash function H is non-trivial, we can not hope to make comparisons to the computational Diffie-Hellman problem without considerable specific knowledge of the properties of H. Rather than consider the security of the scheme with specific choices of H, we assume only that H has negligible collision probability, and instead compare security with the *Decision Diffie-Hellman Problem*. The variant of this problem we consider is as follows. A is given a sequence of tuples, (X_n,Y_n,Z_n,C_n), where X_n,Y_n,Z_n are generated independently at random. With probability 1/2, C_n is the solution to the Diffie-Hellman instance, (X_n,Y_n,Z_n), and with probability 1-1/2=1/2, C_n is generated randomly and independently. A is said to have an $\in$-DDH advantage if A can, with probability 1/2+$\in$, answer the question $\log_{X_n} C_n \overset{?}{=} \log_{X_n} Y_n \log_{X_n} Z_n$.

[0048] Theorem 1, and corollaries 1 and 2 have obvious analogs in the case $H \neq Id$ (assuming only that H has negligible collision probability). Both the statements and proofs are constructed with minor variation, so we only summarize with:

Corollary 3 *Let $\in_{DDH}$ be an upper bound on A's DDH advantage. Then, if H is any hash function with negligible collision probability, an upper bound on the probability that A can submit a vote that differs from the voter's choice, and yet display the correct confirmation* *string is* $\in_0 + (K-1) \in_{DDH}$.

[0049] SVC may not offer any protection if the adversary, A, also controls the vote collection center. If this were the case, A has access to K_i and β_i, and thus can easily display any valid confirmation string of its choosing. It seems unlikely that this would happen, since the vote collection center would be undeniably implicated in the event that such activity is discovered. Nevertheless, in case it is unacceptable to trust the vote collection center in this regard, the "confirmation responsibility" can be distributed among arbitrarily many authorities.

[0050] To distribute the confirmation responsibility, each authority, A_j, $1 \leq j \leq J$, generates (for voter v_i) independent random K_{ij} and β_{ij}. The authorities can combine these by two general methods.

1. **Concatenation**. The voter's confirmation string is computed as a concatenation, in pre-specified order, of the individual confirmation strings (computed separately as in the previous section) corresponding to each of the J authorities. In this case, confirmation is successful only if all of the substrings verify correctly.

2. **Trusted Server or Printer**. If it is acceptable to trust a single central server, or printer, the multiple confirmation strings can be combined into one of the same size by simply computing

$$W_i = \prod_{j=1}^{J} W_{ij} \quad (21)$$

$$U_i = \prod_{j=1}^{J} U_{ij} \quad (22)$$

This has the advantage of reducing the amount of confirmation data that must be transmitted to the voter, but at the cost of creating a central point of attack for the system.

[0051] It is always desirable to reduce the size of the data that must be sent to the voter via the independent channel. As described in section 3, the confirmation dictionary is already small by the standards of modern communications technology, but it may be cost advantageous if even less data can be transmitted. As mentioned above, one approach might be to send the secrets K_i and β_i directly to the voter, but this has the disadvantage of putting a computational burden on the voter that is too large to be executed "in the voter's head," or "on paper." The following variation on the SVC scheme achieves both goals - less data through the independent communication channel, and "mental computation" by the voter. It comes at a cost, namely that the probability that a client adversary may be able to fool the voter is increased, however, this may be quite acceptable from the overall election perspective. Even if the probability of the adversary going undetected is, say 1/2, in order for it to change a substantial fraction of votes, the probability that it will be detected by a statistically significant fraction of voters will be very high. As discussed in the introduction, remedial measures are possible.

[0052] The idea is to deliver the entire set of confirmation strings to the voter via the suspect client, but in randomly permuted order. The only additional piece of information that the voter needs then is the permutation that was used. This isn't quite enough, in this scenario, since all the confirmation strings are available, the adversary can gain some advantage simply by process of elimination. (The case K=2 is particularly useful to consider.) In order to increase the security, we include with the dictionary, several random confirmation strings, that are also permuted.

[0053] The steps in subsection 3.1 are executed as before. In addition, the vote collection sends *to the client*, M_i, a "randomized dictionary," D_i. This is created by the vote collection center, C, as follows:

RD-1. The K (voter specific) confirmation strings

$$\left(S_{i1},\ldots,S_{iK}\right) = \left(H\left(C_{i1}\right),\ldots,H\left(C_{iK}\right)\right) \tag{23}$$

are computed as before.

RD-2. Additionally, L extra strings are generated as

$$\left(S_{i(K+1)},\ldots,S_{i(K+L)}\right) = \left(H\left(g^{e_1}\right),\ldots,H\left(g^{e_L}\right)\right) \tag{24}$$

where the $e_1,\ldots,e_L$ are generated independently at random in Z_q.

RD-3. A random permutation, $\sigma_i \in \Sigma_{K+L}$ is generated.

RD-4. C sets $Q_{ij}=S_{i\sigma(j)}$, for $1 \leq j \leq K+L$, and sets D_i to be the sequence of strings $(Q_{i1}, \ldots Q_{i(K+L)})$.

[0054] If C sends some "human readable" representation of σ_i to v_i, through an independent channel, v_i can now verify her vote by simply finding the confirmation string with the proper index. We denote this scheme by SVCO.

[0055] With respect to the level of security of SVCO, consider the following form of the Diffie-Hellman Decision Problem: A is given a sequence of tuples, $(X_n, Y_n, Z_n, C_n, D_n)$, where X_n, Y_n, Z_n are generated independently at random. Let R_n be generated independently at random, and let O_n be the solution to $\log_{X_n} O_n = \log_{X_n} Y_n \log_{X_n} Z_n$. With probability 1/2, $(C_n, D_n)=(O_n, R_n)$, and with probability 1-1/2=1/2, $(C_n, D_n)=(R_n, O_n)$. A is said to have an $\in$-DDHP advantage if A can, with probability $1/2+\in$, answer the question $\log_{X_n} C_n \overset{?}{=} \log_{X_n} Y_n \log_{X_n} Z_n$. That is, A must answer the same question as in the original version of the problem, but the problem may be easier because more information is available.

[0056] **Theorem 2** *Let $\in_{DDHP}$ be an upper bound on A's DDHP advantage, and H any hash function with negligible collision probability. An upper bound on the probability, under the SVCO scheme, that A can submit a vote that differs from the voter's choice, and yet display the correct confirmation string is*

$$\in_0 + \left(\frac{K+L}{L} \right) \in_{DDHP} \tag{25}$$

[0057] **Proof:** As in the proof of theorem 1, A can simulate an election and SVCO exchange. In this case, however, A must also simulate the list of confirmation strings that were not available in the SVC scheme. For k_1, k_2 fixed, A can pick $C_{ik_1} \in \langle g \rangle$ at random, and for all $k \neq k_2$, pick $\theta_k \in Z_q$ independently at random. A then sets $\mu_K = X^{\theta_k}$. For $k \neq k_1, k_2$, A sets $C_{ik} = C_{ik_1} Y^{\theta_k - \theta_{k_1}}$. A sets $\mu_{k_2} = \mu_{k_1} Z$, and generates L additional random μ_l and l-1 additional C_{il} at random. Finally, A sets $C_{ik_2} = C_{ik_1} C_n$, and the last remaining $C_{il} = C_{ik_1} D_n$. As before, finding the right confirmation string is equivalent to deciding which of the values, C_n, D_n is the correct Diffie-Hellman solution. Averaging over all permutations with uniform probability gives the result.

[0058] Below is described one possible alternative to the secret vote confirmation scheme described above. The level of security between those two schemes is essentially equivalent.

1. In addition to the election public key, h, the vote collection publishes another public key of the form $\bar{h}=h^d$, where $d \in Z_q$ is a secret known only to the vote collection center.

2. The client, M_i, submits a an encrypted ballot on behalf of v_i as before, but *redundantly encrypted* with both h and $\bar{h}$. We denote the second encryption by

$$\left(\bar{X}_i, \bar{Y}_i\right) = \left(g^{\bar{\alpha}_i}, \bar{h}^{\bar{\alpha}_i} m\right) \tag{26}$$

Where $\bar{\alpha}_i$ is selected independently of α_i.

3. M_i also constructs a simple proof of validity (essentially a single Chaum-Pedersen proof) that the two are encryptions of the same value.

4. If the proof of validity does not pass at the vote collection center, corruption is detected as before.

5. The vote collection center selects random $K_i \in \langle g \rangle$; $\beta_i \in Z_q$, and computes

$$T_i = Y_i^{d\beta_i} = \left(h^{\alpha_i}\right)^{d\beta_i} m^{d\beta_i} \tag{27}$$

$$W_i = \bar{Y}_i^{\beta_i} = \left(h^{\bar{\alpha}_i}\right)^{\beta_i} m^{\beta_i} \tag{28}$$

$$V_i = K_i T_i W_i = K_i \bar{h}^{\beta_i(\alpha_i + \bar{\alpha}_i)} m^{(d+1)\beta_i} \tag{29}$$

6. The vote collection center returns $\bar{h}^{\beta_i}$ and V_i to M_i.

7. M_i computes $S_i = K_i m^{(d+1)\beta_i}$ by the equation

$$S_i = K_i m^{(d+1)\beta_i} = \frac{V_i}{(\bar{h}^{\beta_i})^{(\alpha_i + \bar{\alpha}_i)}} \tag{30}$$

and displays this value (or, $H(S_i)$) to the voter, v_i.

8. The voter requests a confirmation dictionary as before, and checks against the displayed value.

[0059] In the case of detected corruption, corrective action is taken as before.

[0060] The description of the facility above describes using a single d (and therefore a single $\bar{h}=h^{d}$) for all voters and publishing this value in advance of the election.

[0061] Alternatively, the vote collection center (or distributed set of "confirmation authorities") issues an independent, random d_i (and therefore $\bar{h}_i=h^{d_i}$) for each voter, v_i. The value d_i is always kept secret, but the value $\bar{h}_i$ is communicated to v_i.

[0062] In one embodiment, the facility communicates $\bar{h}_i$ to v_i as follows:

A-1 v_i contacts the vote collection center and authenticates himself/herself

A-2 Assuming authentication is successful, the vote collection center:

1. Generates d_i randomly
2. Computes $\bar{h}_i=h^{d_i}$
3. Sends $\bar{h}_i$ to v_i

A-3 The voter, v_i then proceeds as described above with $\bar{h}_i$ in place of $\bar{h}$

[0063] In another embodiment, the facility communicates $\bar{h}_i$ to v_i as follows:

B-1 v_i contacts vote collection center (and optionally authenticates himself/herself)

B-2 v_i makes ballot choice m_i, and returns the encrypted ballot $(g^{\alpha_i},h^{\alpha_i}m_i)$

B-3 The vote collection center *at this point*:

1. Generates d_i randomly
2. Computes $\bar{h}_i=h^{d_i}$
3. Sends $\bar{h}_i$ to v_i

B-4 Voter, v_i then

1. Generates second encryption of m_i as $(g^{\bar{\alpha}_i},\bar{h}_i^{\bar{\alpha}_i}m_i)$
2. Generates same *proof of validity* showing that first and second encryptions are encryptions of the same ballot choice, m_i
3. Sends both the second encryption, and the proof of validity to the ballot collection agency

B-5 The rest of the confirmation process proceeds as described above

[0064] Figures 1-3 illustrate certain aspects of the facility. Figure 1 is a high-level block diagram showing a typical environment in which the facility operates. The block diagram shows several voter computer systems 110, each of which may be used by a voter to submit a ballot and verify its uncorrupted receipt. Each of the voter computer systems are connected via the Internet 120 to a vote collection center computer system 150. Those skilled in the art will recognize that voter computer systems could be connected to the vote collection center computer system by networks other than the Internet, however. The facility transmits ballots from the voter computer systems to the vote collection center computer system, which returns an encrypted vote confirmation. In each voter computer system, the facility uses this encrypted vote confirmation to determine whether the submitted ballot has been corrupted. While preferred embodiments are described in terms in the environment described above, those skilled in the art will appreciate that the facility may be implemented in a variety of other environments including a single, monolithic computer system, as well as various other combinations of computer systems or similar devices connected in various ways.

[0065] Figure 2 is a block diagram showing some of the components typically incorporated in at least some of the computer systems and other devices on which the facility executes, such as computer systems 110 and 130. These computer systems and devices 200 may include one or more central processing units ("CPUs") 201 for executing computer programs; a computer memory 202 for storing programs and data while they are being used; a persistent storage device 203, such as a hard drive for persistently storing programs and data; a computer-readable media drive 204, such as a CD-ROM drive, for reading programs and data stored on a computer-readable medium; and a network connection 205 for connecting the computer system to other computer systems, such as via the Internet. While computer systems configured as described above are preferably used to support the operation of the facility, those skilled in the art will appreciate that the facility may be implemented using devices of various types and configurations, and having various components.

[0066] Figure 3 is a flow diagram showing steps typically performed by the facility in order to detect a compromised ballot. Those skilled in the art will appreciate that the facility may perform a set of steps that diverges from those shown, including proper supersets and subsets of these steps, reorderings of these steps, and steps of sets in which performance of certain steps by other computing devices.

[0067] In step 301, on the voter computer system, the facility encodes a ballot choice selected by the voter in order to form a ballot. In step 302, the facility encrypts this ballot. In some embodiments, the encrypted ballot is an ElGamal pair, generated using an election public key and a secret maintained on the voter computer system. In step 303, the facility optionally signs the ballot with a private key belonging to the voter. In step 304, the facility constructs a validity proof that demonstrates that the encrypted ballot is the encryption of a ballot in which a valid ballot choice is selected. In step 305, the facility transmits the encrypted, signed ballot and the validity proof to a vote collection center computer system.

[0068] In step 321, the facility receives this transmission in the vote collection center computer system. In step 322, the facility verifies the received validity proof. In step 323, if the validity proof is successfully verified, then the facility continues with 324, else the facility does not continue in step 324. In step 324, the facility generates an encrypted confirmation of the encrypted ballot. The facility does so without decrypting the ballot, which is typically not possible in the vote collection center computer system, where the secret used to encrypt the ballot is not available. In step 325, the facility transmits the encrypted confirmation 331 to the voter computer system.

[0069] In step 341, the facility receives the encrypted vote confirmation in the voter computer system. In step 342, the facility uses the secret maintained on the voter computer system to decrypt the encrypted vote confirmation. In step 343, the facility displays the decrypted vote confirmation for viewing by the user. In step 344, if the displayed vote confirmation is translated to the ballot choice selected by the voter by a confirmation dictionary in the voter's possession, then the facility continues in step 345, else the facility continues in step 346. In step 345, the facility determines that the voter's ballot is not corrupted, whereas, in step 346, the facility determines that the voter's ballot is corrupted. In this event, embodiments of the facility assist the user in revoking and resubmitting the voter's ballot.

[0070] It will be appreciated by those skilled in the art that the above-described facility may be straightforwardly adapted or extended in various ways. While the foregoing description makes reference to preferred embodiments, the scope of the invention is defined solely by the claims that follow and the elements recited therein.